UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/650,245	08/27/2003	Wolfgang Grieskamp	3382-64897	7189

26119 7590 01/03/2007

KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET
SUITE 1600
PORTLAND, OR 97204

EXAMINER	
PHAM, THAI V	
ART UNIT	PAPER NUMBER
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	01/03/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	**Application No.** 10/650,245	**Applicant(s)** GRIESKAMP ET AL.	
	Examiner Thai Van Pham	**Art Unit** 2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 August 2003*.

2a)□ This action is **FINAL**. 2b)☒ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-21* is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *27 August 2003* is/are: a)☒ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)□ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____.

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date *05/20/2004*.

4) □ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) □ Notice of Informal Patent Application

6) □ Other: _____.

DETAILED ACTION

This is the initial office action based on the application filed on 08/27/2003.

Priority date that has been considered for this application is 08/27/2003.

Claims 1 – 21 are currently pending and have been considered below.

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

1. Claims 1 – 6 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

-- Claim 1.

Claim 1 recites "A computer readable medium containing a data structure..." as the claimed subject matter. The claim further recites "the computer readable medium comprising..." the elements stated in the body of the claim. The data structure associated with the claimed computer product does not impart any functionality when employed as a computer element. Furthermore, the elements of the claim do not define any "acts" being performed by the associated data structure contained in the claimed computer readable medium. Thus, the claim computer product does not provide any tangle and concrete result.

See MPEP 2106.01: "Descriptive material can be characterized as either "functional descriptive material" or "nonfunctional descriptive material." In this context, "functional descriptive material" consists of data structures and computer programs which impart functionality when employed as a computer component. (The definition of "data structure" is "a physical or logical relationship among data elements, designed to support specific data manipulation functions." The New IEEE Standard Dictionary of Electrical and Electronics Terms 308 (5th ed. 1993).)) ... ".

-- Claims 2 – 6.

Claims 2 – 6 are dependent claims of claim 1. These claims all fail to remedy the 35 USC 101 nonstatutory problem of claim.1.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

2. Claims 7 – 9 and 12 – 20 are rejected under 35 U.S.C. 102(e) as being anticipated by **Bates et al.** (US 2002/0174416).

-- Claim 7.

Bates discloses *a computerized method comprising*:

- *receiving via an application programming interface a request to create a state save*;

(Fig. 5, items 518 and 526 – "Variable to monitor" and "Condition to take snap shots" – and associated text, e.g., [0046] and [0047].)

• *saving a first representation of a state of an executing program in response to the request*;

(Fig. 3B and associated text, e.g., [0041]; "...there is one snapshot or table of occurrence each time the trigger variable/expression A **312** is referenced or written or read by the program ... ". In the context of snapshots being created in succession with the program execution, the first snapshot is hereby regarded as <u>first state frame</u> of all snapshots of a particular trigger variable.)

• maintaining a second representation of subsequent state comprising changes made to the state of the executing program after the first representation; and

(Fig. 3B and associated text, e.g., [0041]; "...Block **364** points to another entry in the history table which could be another table of occurrence of similar structure of that shown in **Fig. 3b**." The snapshot after the first snapshot is hereby regarded as <u>second state frame</u>.)

• resetting the executing program to the saved first representation upon receiving a state set request at the application programming interface.

(Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044]. From any state, the program/model can be reset back to any other saved states.)

-- <u>Claim 8</u>.

Bates discloses *a computer system comprising memory and a central processing unit* (Fig. 2 – "Processor" and "Memory") *executing*,

- *a program including executable instructions and an evolving present state*; and ([0038]: "...record a history of a computer expression through an executing computer program or process ... ".)
- a state component comprising:

  - an initial representation of a prior evolving present state of the program, (Fig. 3B and associated text, e.g., [0041]; "...there is one snapshot or table of occurrence each time the trigger variable/expression A **312** is referenced or written or read by the program ... ". In the context of snapshots being created in succession with the program execution, the first snapshot is hereby regarded as <u>initial state representation</u> of all snapshots of a particular trigger variable.)
  - a subsequent representation of state changes made by the program since the initial representation, and (Fig. 3B and associated text, e.g., [0041]; "...Block **364** points to another entry in the history table which could be another table of occurrence of similar structure of that shown in **Fig. 3b**." The snapshot after the first snapshot is hereby regarded as <u>subsequent state representations</u>.)
  - a method for returning the program state to the prior evolving present state. (Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044]. From any state, the program/model can be reset back to any other saved states.)

-- <u>Claim 9</u>.

Bates discloses *the computer system of claim 8 wherein*

- *the state component further includes a method for reading a location and value from the initial representation of the prior evolving present state and storing the value in a subsequent representation of the state changes.*

(Fig. 3B and associated text, e.g., [0041]; "...Block **364** points to another entry in the history table which could be another table of occurrence of similar structure of that shown in **Fig. 3b**." As the program execution progress over time, successive encounters of the same trigger variable will result in snapshots in which some of the variables may change and others may not. In the case when variables retain the same values between successive snapshots, they are basically transferred from state to the other.)

-- Claim 12.

Bates discloses *a method comprising*:

- *executing a program which executes an executable model comprising a transforming present model state and actions causing changes to the present model state*;

([0038]: "...record a history of a computer expression through an executing computer program or process ... ".)

- *saving a present state of the executable model in response to a state save request received from the program via an application programming interface*;

(Fig. 5, items 518 and 526 – "Variable to monitor" and "Condition to take snap shots" – and associated text, e.g., [0046] and [0047].)

- *setting the present model state of the executable model to the saved present state in response to a state set request received from the program via the application programming interface*; and

(Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044]. The state of the program/model can be reset back to any saved states.)

- *after setting the present model state, executing another action causing changes to the set present model state.*

(Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044]. From any state, the program/model can be reset back to any other saved states.)

-- Claim 13.

Bates discloses *the method of claim 12 wherein*

- *the executing model exercises plural actions, each action exercised after setting the present model state to the saved present state.*

(Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044]. Subsequent execution of the program from a specific state consists of a collection of actions as instructed by the program's instructions.)

-- Claim 14.

Bates discloses *the method of claim 13 wherein*

- *the plural actions are recorded thereby creating action sequences used to exercise a state space of a program that the executable model is designed to model.*

(Fig. 4, items 442, 444a, 444b, 444c and associated text, e.g., [0044]. Snapshots of a particular trigger variable and a collection of snapshots of all trigger variables constitute a state space of the program under execution.)

-- Claim 15.

Bates discloses *a computer system comprising memory and a central processing unit* (Fig. 2 – "Processor" and "Memory") *and executing*

- *a program comprising instructions and a state changing relative to time, and*

(Fig. 3B and associated text, e.g., [0041]; "...there is one snapshot or table of occurrence each time the trigger variable/expression A **312** is referenced or written or read by the program ... ".)

- *a state mechanism linked into the program and providing*

  - a save state reference the program accesses to save a present changing state,

  (Fig. 5, items 518 and 526 – "Variable to monitor" and "Condition to take snap shots" – and associated text, e.g., [0046] and [0047].)

  - *a service that creates a current state record and saves a representation of state changes made by the program after the program saves a present changing state via the reference, and*

  (Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044].)

  - *a state set reference the program accesses to reset the program state to a prior saved present changing state.*

  (Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044]. From any state, the program/model can be reset back to any other saved states.)

-- Claim 16.

Bates discloses *the system of claim 15*, wherein

- *the state mechanism further provides a service that reads a value stored in a prior saved state and copies the read value into the current state record.*

(Fig. 3B and associated text, e.g., [0041]; "...Block **364** points to another entry in the history table which could be another table of occurrence of similar structure of that shown in **Fig. 3b**." As the program execution progress over time, successive encounters of the same trigger variable will result in snapshots in which some of the variables may change and others may not. In the case when variables retain the same values between successive snapshots, they are basically transferred from state to the other.)

-- Claim 17.

Claim 17 is a computer product claim ([0033]: ""...The debugger software application **50** is resident in memory **32** for the purpose of debugging one or more executable computer programs...") for performing a method corresponding to the method of claim 7. Therefore, claim 17 is rejected for the same reason set forth in connection to the rejection of claim 7 above.

-- Claim 18.

Claim 18 is a computer product claim ([0033]: ""...The debugger software application **50** is resident in memory **32** for the purpose of debugging one or more executable computer programs...") for performing a method corresponding to the method of claim

12. Therefore, claim 18 is rejected for the same reason set forth in connection to the rejection of claim 12 above.

-- <u>Claim 19</u>.

Bates discloses *a computer readable method comprising computer executable instructions for performing a method comprising*:

- *receiving a request to create a saved state of an executing model*;

(Fig. 5, items 518 and 526 – "Variable to monitor" and "Condition to take snap shots" – and associated text, e.g., [0046] and [0047].)

- *saving a first representation of a state of the executing model*;

(Fig. 3B and associated text, e.g., [0041]; "...there is one snapshot or table of occurrence each time the trigger variable/expression A **312** is referenced or written or read by the program ... ". In the context of snapshots being created in succession with the program execution, the first snapshot is hereby regarded as <u>first state representation</u> of all snapshots of a particular trigger variable.)

- *maintaining a second representation of state changes made by the executing model after the first representation*; and

(Fig. 3B and associated text, e.g., [0041]; "...Block **364** points to another entry in the history table which could be another table of occurrence of similar structure of that shown in **Fig. 3b**." The snapshot after the first snapshot is hereby regarded as <u>second state representation</u>.)

- *reinstating the executing model state to the state of the first representation.*

(Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044]. From any state, the program/model can be reset back to any other saved states.)

-- Claim 20.

Bates discloses *a computerized method comprising*:

- *executing a program comprising a transforming present state*;

(Fig. 3B and associated text, e.g., [0041]; "...there is one snapshot or table of occurrence each time the trigger variable/expression A **312** is referenced or written or read by the program ... ".)

- *saving a present state of the program in response to a state save request received from the program via an application programming interface*; and

(Fig. 5, items 518 and 526 – "Variable to monitor" and "Condition to take snap shots" – and associated text, e.g., [0046] and [0047].)

- *after the program transforms state from the saved present state, setting the present state of the program to the saved present state in response to a state set request received from the program via the application programming interface.*

(Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044]. From any state, the program/model can be reset back to any other saved states.)

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1 and 6 are rejected under 35 U.S.C. 103(a) as being obvious over **Bates et al.** (US 2002/0174416).

-- Claim 1.

Bates discloses *a computer readable medium containing a data structure* ([0033]: ""...The debugger software application **50** is resident in memory **32** for the purpose of debugging one or more executable computer programs...") *for saving state for a semantically accessible state binding method* (abstract: snapshots of a trigger variable and associated variables and expressions), *the computer readable medium comprising*:

- *a first state frame including a representation of a state of an executing program*; and (Fig. 3B and associated text, e.g., [0041]; "...there is one snapshot or table of occurrence each time the trigger variable/expression A **312** is referenced or written or read by the program ... ". In the context of snapshots being created in succession with the program execution, the first snapshot is hereby regarded as first state frame of all snapshots of a particular trigger variable.)
- *a second state frame including a representation of state changes made by the executing program after the first state frame is created* and *the first state frame includes a pointer to the second state frame.*

(Fig. 3B and associated text, e.g., [0041]; "...Block **364** points to another entry in the history table which could be another table of occurrence of similar structure of that shown in **Fig. 3b**." The snapshot after the first snapshot is hereby regarded as second state frame. The first state frame includes a pointer pointing to the second state frame.)

Bates does not explicitly disclose that

- *the second state frame includes a pointer back to the first state frame.*

However, **Bates** discloses that the different snapshots of a particular trigger variable are recorded and accessible to the user at anytime (Fig. 4, e.g., items 444a, 444b, 444c and associated text, e.g., [0044] – [0045]). Using the user interface, the user can traverse between different snapshot instances of the program execution in any order.

Therefore, It would have been obvious to one of ordinary skill in the art at the time the invention was made to include another backward pointer in the second snapshot which points back to the first snapshot, in addition to the forward pointer which points to the following snapshot, in **Bates**' disclosure in order to create a doubly linked list as a way to make any snapshot of a particular trigger variable accessible to the user in both forward and backward traversals.

-- Claim 6.

Bates discloses *the computer readable medium of claim 1 wherein*

- *the second state frame includes unchanged state read from the first state frame.*

(Fig. 3B and associated text, e.g., [0041]; "...Block **364** points to another entry in the history table which could be another table of occurrence of similar structure of that shown in **Fig. 3b**." As the program execution progress over time, successive encounters of the same trigger variable will result in snapshots in which some of the variables may change and others may not. In the case when variables retain the same values between successive snapshots, they are basically transferred from one state to another.)

4. Claims 2 – 5, 10, 11, and 21 are rejected under 35 U.S.C. 103(a) as being obvious over **Bates et al.** (US 2002/0174416) in view of **Vishkin** (US 6,463,527).

-- Claim 2.

Bates discloses *the computer readable medium of claim 1*, however, does not explicitly disclose the method further comprising:

- *a third state frame including a representation of state changes made by the executing program after a fork method creates the third frame*, and

Vishkin discloses a method for spawning program instructions into currently running threads (Fig. 2, items 20 and 22 and associate text, e.g., Cols. 3 and 4). The method of **Bates** for monitoring program execution states implicitly includes sequential as well as concurrent executions of processes within a program. In the case of splitting a program sequential execution (Fig. 2, item 20) into multiple concurrent execution (Fig. 2, item 22) of processes as disclosed by **Vishkin**, the execution states, in parallel, of the program immediately following a splitting point, or fork, are hereby considered third state frame after a fork.

Therefore, it would have obvious to one of ordinary skill in the art at the time the invention was made to recognize that in computing environment where parallel executions of processes in a program are detected and allowed as disclosed in **Vishkin**, when a trigger variable is encountered in these concurrently executed threads, the method of **Bates** takes snapshots of the program state at these points in order to properly trace the program execution history during the execution of these concurrent processes.

• *the third state frame includes a pointer back to the second frame.*

However, **Bates** discloses that the different snapshots of a particular trigger variable are recorded and accessible to the user at anytime (Fig. 4, e.g., items 444a, 444b, 444c and associated text, e.g., [0044] – [0045]). Using the user interface, the user can traverse between different snapshot instances of the program execution in any order.

Therefore, It would have been obvious to one of ordinary skill in the art at the time the invention was made to include another backward pointer in the second snapshot which points back to the first snapshot, in addition to the forward pointer which points to the following snapshot, in **Bates**' disclosure in order to create a doubly linked list as a way to make any snapshot of a particular trigger variable accessible to the user in both forward and backward traversals.

-- Claim 3.

Bates and **Vishkin** disclose *the computer readable medium of claim 2*, and

Vishkin further discloses that

• *a fourth frame includes changes made by the executing program after the fork method creates the third state frame*, and

(Fig. 2, item 22 and associated text, e.g., Col. 3 and 4. Snapshots of trigger variables running in parallel to the third frame belonging to other concurrently executed threads are hereby considered fourth frame(s).)

Bates further discloses that

• *after a set method returns the executing program to the state of the second state frame.*

(Fig. 4, items 444a, 444b, 444c and associated text, e.g., [0044]. From any state, the program/model can be reset back to any other saved states.)

-- Claim 4.

Bates and **Vishkin** disclose *the computer readable medium of claim 3*, and **Vishkin** further discloses the method further comprising

• *a joined state frame including a combination of state changes in the third and fourth frames.*

(Fig. 2, items 23 and 24 and associated text, e.g., Col. 3 and 4. Snapshots of trigger variables following the concurrently executed third and fourth frames after being joined into serial execution.)

-- Claim 5.

Bates and **Vishkin** disclose *the computer readable medium of claim 3*, **Bates** further discloses the method comprising

• *a first thread of the executing program makes state changes copied in the second frame*, and

(Fig. 3B and associated text, e.g., [0041]; "...Block **364** points to another entry in the history table which could be another table of occurrence of similar structure of that shown in **Fig. 3b**." As the program execution progress over time, successive encounters of the same trigger variable will result in snapshots in which some of the variables may change and others may not. In the case when variables retain the same values between successive snapshots, they are basically transferred from state to the other.)

- a second thread of the executing program makes state changes copied into the third frame.

(Fig. 3B and associated text, e.g., [0041. In concurrent execution of program processes, when a trigger variable is encountered in one or more threads, the unchanged associated variables are copied from the second frame to third frame(s) and the changes of the trigger variables and other associated variables are correspondingly recorded.)

-- Claim 10.

Bates discloses *the computer system of claim 8*, however, does not explicitly disclose that

- *the state component includes a fork method for maintaining state for a thread spawned by the program and a forked representation of state changes made by the spawned thread of the program.*

Vishkin discloses a method for spawning program instructions into currently running threads (Fig. 2, items 20 and 22 and associate text, e.g., Cols. 3 and 4). The method of **Bates** for monitoring program execution states implicitly includes sequential as well as concurrent executions of processes within a program. In the case of splitting a program sequential execution (Fig. 2, item 20) into multiple concurrent execution (Fig. 2, item 22) of processes as disclosed by **Vishkin**, the execution states, in parallel, of the program immediately following a splitting point, or fork, are hereby considered third state frame after a fork.

Therefore, it would have obvious to one of ordinary skill in the art at the time the invention was made to recognize that in computing environment where parallel executions of processes in a program are detected and allowed as disclosed in **Vishkin**, when a trigger variable is encountered in these concurrently executed threads, the method of **Bates** takes snapshots of the program state at these points in order to properly trace the program execution history during the execution of these concurrent processes.

-- Claim 11.

Bates and **Vishkin** disclose *the system of claim 10* and Vishkin further discloses

- *the state component includes a join method for joining state changes made by the forked thread back into state changes of the subsequent representation.*

(Fig. 2, items 23 and 24 and associated text, e.g., Col. 3 and 4. Snapshots of trigger variables following the concurrently executed processes after being joined into serial execution.)

-- Claim 21.

Bates discloses *the method of claim 20*, however, does not explicitly disclose that

- *the program varies an execution path after setting the present state to the saved present state.*

Vishkin discloses a method for spawning program instructions into currently running threads, or allowing the program to vary its execution path from a fork point (Fig. 2, items 20 and 22 and associate text, e.g., Cols. 3 and 4). The method of **Bates** for monitoring program execution states implicitly includes sequential as well as concurrent executions of processes within a program. In the case of splitting a program sequential execution (Fig. 2, item 20) into multiple concurrent execution (Fig. 2, item 22) of processes as disclosed by **Vishkin**, the execution states, in parallel, of the program immediately following a splitting point, or fork, are hereby considered <u>third state frame</u> after a fork.

Therefore, it would have obvious to one of ordinary skill in the art at the time the invention was made to recognize that in computing environment where parallel executions of processes in a program are detected and allowed as disclosed in **Vishkin**, when a trigger variable is encountered in these concurrently executed threads, the method of **Bates** takes snapshots of the program state at these points in order to properly trace the program execution history during the execution of these concurrent processes.

Conclusion

The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure. See the attached Notice of References Cited.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Thai Van Pham whose telephone number is (571) 270-1064. The examiner can normally be reached on Monday - Thursday, 8am - 3pm EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TVP

TUAN DAM
SUPERVISORY PATENT EXAMINER